Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A computer system configured for providing themes for ~~graphical components~~ controls of a first application and a second application in a graphical operating system environment, the computer system having memory, the computer system comprising:

a selecting module receiving a user request for a selected theme having an associated non-binary theme file with theme properties ~~capable of being applied to one or more applications in the graphical operating system environment~~, the theme properties of the non-binary theme file capable of being applied to the controls of the first application, the theme properties of the non-binary file incapable of being applied to the controls of the second application;

a converting module converting the associated non-binary theme file into a shared binary theme file to facilitate retrieval of theme properties; and

a loading module loading the shared binary theme file into the memory so that themes can be applied to the ~~graphical~~ controls of the first application and the second application.

2. (Currently Amended) The system of claim 1 further comprising a plurality of processes, each process accessing the shared binary theme file.

3. (Currently Amended) The system of claim 1, further comprising:

an update handle module receiving a theme handle request from a ~~graphical component~~ control and distributing a theme handle if the ~~graphical component~~ control is found in the shared binary theme file so that the ~~graphical component~~ control can use the theme properties of the shared binary theme file; and

a close handle module closing the theme handle and decrementing a reference count on the shared memory in response to process termination so that a theme handle can be closed when a shared binary theme file is loaded.

4. (Currently Amended) The system of claim 1, further comprising:

a notification module notifying the processes that a new shared theme file has been loaded.

5. (Currently Amended) The system of claim 1, wherein the converting module comprises:

a schema file parsing module parsing a schema file containing a list of all themeable ~~graphical components~~ controls and properties;

a theme specification file parsing module parsing a theme specification file specifying ~~graphical component~~ control sizes and colors; and

a building module building a shared binary theme file containing the ~~graphical components~~ controls, properties, sizes, and colors in a binary format.

6. (Original) The system of claim 5 wherein the binary format is hierarchical, there being a data section for each hierarchy, the sections being a global section, a class section, a parts section, and a states section.

7. (Original) The system of claim 6 wherein the converting module further builds a packed data object section having all the theme properties for a class, part, and state.

8. (Currently Amended) A method for creating a visual style for a set of ~~graphical components~~ controls of a first application and a second application for use on a computer system having a graphical operating system environment and processes with shared memory, the method comprising:

selecting ~~graphical components~~ controls, from a schema file of ~~graphical components~~ controls, that are desired to have a defined visual style, each ~~component~~ control being defined by a unique class name;

assigning properties to the selected ~~components~~ controls according to the defined visual style so that each selected ~~component~~ control has assigned properties;

grouping the pairs of selected ~~graphical components~~ controls and corresponding assigned properties for the defined visual style together in a class data file, the first application operable to read the class data file;

converting the class data file into a shared binary theme file having a class data section having class names and assigned properties in a binary format, the first application and the second application both operable to read the shared binary theme file; and

loading the shared binary theme file into the shared memory so that a visual style can be used to render ~~graphical components~~ controls ~~of a plurality of applications~~ in the graphical operating system environment.

9. (Currently Amended) The method of claim 8, wherein the ~~graphical components~~ controls defined within the schema file of ~~graphical components~~ controls have one or more part names associated with at least one class name, and the converting act further comprises creating a part property data section in the shared binary theme file, the part property data section having the one or more part names and the assigned properties.

10. (Currently Amended) The method of claim 8, wherein the ~~graphical components~~ controls defined within the schema file of ~~graphical components~~ controls have one or more state names associated with at least one defined part name, and the converting act further comprises creating a state property data section in the shared binary theme file, the state property data section having the one or more state names and the assigned properties.

11. (Currently Amended) The method of claim 8 further comprising:

identifying some properties as global properties; and

creating in the shared binary theme file a global properties section having the global properties to be used when a class name, part name, or state name cannot be found in the shared binary theme file.

12. (Currently Amended) The method of claim 11, wherein a list of available properties is within the first schema file of ~~graphical components~~ controls, that may be selected in the selecting step for each ~~graphical component~~ control, part and state.

13. (Currently Amended) The method of claim 12, wherein the act of converting comprises:

identifying a derived property for a ~~graphical component~~ control;

associating a unique numeric identifier with the derived property to create a derived property identifier;

identifying one or more primitive properties for each derived property, wherein each primitive property has associated property data having a length;

associating a unique numeric identifier with each primitive property, to create a primitive property identifier;

calculating the lengths of each of the associated property data;

selecting a derived property identifier;

writing a binary tagged data module to a tagged data memory offset in the class data section of the shared binary file wherein the binary tagged data module contains the selected derived property identifier, the one or more primitive property identifiers, the associated property values, and each of the property values' lengths; and

writing an associated parent part offset after each binary tagged data module, the associated parent part offset being a memory offset into the global class section.

14. (Currently Amended) The method of claim 13 wherein the act of converting further comprises:

obtaining the memory offset of a binary tagged data module for a state; and

writing the memory offset to a second memory offset in a state jump table in the shared binary theme file.

15. (Currently Amended) The method of claim 14 wherein the act of converting further comprises:

writing the second memory offset to a third memory offset in a part jump table in the shared binary theme file.

16 - 20. Canceled.

21. (Currently Amended) A computer program product, stored on at least one tangible media, readable by a computing system and encoding a computer program of instructions for executing a computer process for creating a visual style for a set of ~~graphical components~~ controls of a first application and a second application for use on a computer system

having a graphical operating system environment and processes with shared memory, said computer process comprising:

selecting ~~graphical components~~ controls, from a schema file of ~~graphical components~~ controls, that are desired to have a defined visual style, each ~~component~~ control being defined by a unique class name;

assigning properties to the selected ~~components~~ controls according to the defined visual style so that each selected ~~component~~ control has assigned properties;

grouping the pairs of selected ~~graphical components~~ controls and corresponding assigned properties for the defined visual style together in a class data file, the first application operable to read the class data file;

converting the class data file into a shared binary theme file having a class data section having class names and assigned properties in a binary format, the first application and the second application both operable to read the shared binary theme file; and

loading the shared binary theme file into the shared memory so that a visual style can be used to render ~~graphical components~~ controls ~~of a plurality of applications~~ in the graphical operating system environment.

22. (Currently Amended) The computer program product of claim 21 wherein the ~~graphical components~~ controls defined within the schema file of ~~graphical components~~ controls have one or more part names associated with at least one class name, and the converting act further comprises creating a part property data section in the shared binary theme file, the part property data section having the one or more part names and the assigned properties.

23. (Currently Amended) The computer program product of claim 22 wherein the ~~graphical components~~ controls defined within the schema file of ~~graphical components~~ controls have one or more state names associated with at least one defined part name, and the converting act further comprises creating a state property data section in the shared binary theme file, the state property data section having the one or more state names and the assigned properties.

24. (Currently Amended) The computer program product of claim 21 further comprising:

identifying some properties as global properties; and

creating in the shared binary theme file a global properties section having the global properties to be used when a class name, part name, or state name cannot be found in the shared binary theme file.

25. (Currently Amended) The computer program product of claim 24, wherein a list of available properties is within the first schema file of ~~graphical components~~ controls, that may be selected in the selecting step for each ~~graphical component~~ control, part and state.

26. (Currently Amended) The computer program product of claim 22, wherein the act of converting comprises:

identifying a derived property for a ~~graphical component~~ control;

associating a unique numeric identifier with the derived property to create a derived property identifier;

identifying one or more primitive properties for each derived property, wherein each primitive property has associated property data having a length;

associating a unique numeric identifier with each primitive property, to create a primitive property identifier;

calculating the lengths of each of the associated property data;

selecting a derived property identifier;

writing a binary tagged data module to a tagged data memory offset in the class data section of the shared binary file wherein the binary tagged data module contains the selected derived property identifier, the one or more primitive property identifiers, the associated property values, and each of the property values' lengths; and

writing an associated parent part offset after each binary tagged data module, the associated parent part offset being a memory offset into the global class section.

27. (Currently Amended) The computer program product of claim 26, wherein the act of converting further comprises:

obtaining the memory offset of a binary tagged data module for a state; and

writing the memory offset to a second memory offset in a state jump table in the shared binary theme file.

28. (Currently Amended) The computer program product of claim 27 wherein the act of converting further comprises writing the second memory offset to a third memory offset in a part jump table in the shared binary theme file.

29 - 33. Canceled.

34. (New) A method executed on a computer system for providing themes for controls of a first application and a second application in a graphical operating system environment, method comprising:

receiving a first render request for a first control of the first application, the first control capable of being rendered according to properties in a non-binary theme file;

receiving a second render request for a second control of the second application, the second control incapable of being rendered according to properties in a non-binary theme file;

converting the non-binary theme file to a shared binary theme file;

in response to the first render request, accessing the shared binary theme file to retrieve theme property data for the first control;

in response to the second render request, accessing the shared binary theme file to retrieve theme property data for the second control; and

rendering the first and second control according to the retrieved theme property data.

35. (New) The method of claim 34, wherein the first and second render requests include a theme handle and a component state.

36. (New) The method of claim 34, wherein the act of accessing the shared binary theme file comprises:

retrieving an offset into a class data section of the shared binary theme file, the class data section having theme property data for a class in binary format;

performing a binary search for class property data at the offset;

determining if class property data exists at the offset;

jumping to a global data section of the shared binary theme file having global theme property data, if no class property data is found; and

retrieving global theme property data from the global data section.

37. (New) The method of claim 34, wherein the act of accessing the shared binary theme file comprises:

retrieving an offset into a part jump table section of the shared binary theme file, the part jump table section having theme property data for a part in binary format;

performing a binary search for part property data at the offset;

determining if part property data exists at the offset;

jumping to a class data section of the shared binary theme file having theme property data for a class, if no part property data is found; and

retrieving class theme property data from the class data section.

38. (New) The method of claim 34, wherein the step of accessing the shared binary theme file comprises:

retrieving a memory offset into a part jump table section of the shared binary theme file;

retrieving from the part jump table section a second memory offset into a state jump table section;

jumping to the second memory offset of the shared binary theme file having state theme property data; and

retrieving state theme property data from the state theme property data section.